Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 144 (2004) 2–16

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

On data classification by iterative linear partitioning

Martin Anthony

Department of Mathematics, The London School of Economics and Political Science, Houghton Street, London WC2A 2AE, UK

Received 5 January 2003; received in revised form 17 April 2004; accepted 24 April 2004

Abstract

We analyze theoretically the generalization properties of multi-class data classification techniques that are based on iteratively partitioning the data points by hyperplanes. A special case is that in which the data points of different classes are separated by a number of parallel hyperplanes, and we investigate the algorithmics of finding a suitable partitioning in this case.

Keywords: Pattern classification; Multi-class classification; Hyperplanes; Decision lists

1. Introduction

Suppose that we have been given some data points in $\mathbb{R}^n$, each with a *classification*, which we shall assume to be a non-negative integer in some range $[k] = \{0, 1, \ldots, k\}$ where $k \in \mathbb{N}$. The data points, together with their classifications will be denoted D. An *extension* of D is a function $f : \mathbb{R}^n \to [k]$ such that f agrees with D; that is, if x is one of the data points given in D then $f(x) = j$ if and only if x is classified with label j in D (for any $j \in [k]$). A common aim in machine learning and data mining is to find an extension of f which will, in a sense to be made precise, be a good 'generalization' of the data. By this we mean that we should like it to be the case that for most points that are not in D, the extension f classifies y correctly. We might also consider *partial extensions*, by which we mean functions that agree with a large proportion—though not necessarily all—of the classifications of the points in D.

There are clearly very many extensions of a given data set. We shall analyze the performance of methods based on the use of *threshold decision lists* and (a special subclass of these) *multivalued multithreshold functions*. Such methods arise when the data points are iteratively separated by hyperplanes. In quantifying the performance of such methods, we employ a probabilistic framework that has been used extensively in the modeling of machine learning; see the books [5,6,41,42], for example.

Section 2 describes the classification techniques that are the focus of this paper. Section 3 discusses how generalization error can be modeled, and presents the results, with the proofs given in Section 4. Section 5 discusses algorithmic issues for finding multivalued multithreshold functions, and raises some open questions.

2. Threshold decision lists

2.1. Decision lists

We start by describing binary-valued *decision lists*, introduced by Rivest [36]. Suppose that G is any set of functions from some set X to $\{0, 1\}$. We shall usually suppose (for the sake of simplicity) that G contains the identically-1 function. A function $f : X \to \{0, 1\}$ is said to be a *decision list* based on G if for some positive integer r, there are functions $f_1, f_2, \ldots, f_r \in G$ and

E-mail address: m.anthony@lse.ac.uk (M. Anthony).

0166-218X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2004.04.003

bits $c_1, c_2, \ldots, c_r \in \{0, 1\}$ such that f acts as follows. Given an example y, we first evaluate $f_1(y)$. If $f_1(y) = 1$, we assign value c_1 to $f(y)$; if not, we evaluate $f_2(y)$, and if $f_2(y) = 1$ we set $f(y) = c_2$, otherwise we evaluate $f_3(y)$, etc. If y fails to satisfy any f_i then $f(y)$ is given the default value 0. The evaluation of a decision list f can therefore be thought of as a sequence of '`if then else`' commands, as follows:

`if` $f_1(y) = 1$ `then` *set* $f(y) = c_1$
`else if` $f_2(y) = 1$ `then` *set* $f(y) = c_2$
$\ldots$
$\ldots$
`else if` $f_r(y) = 1$ `then` *set* $f(y) = c_r$
`else` *set* $f(y) = 0$.

Clearly (as in [10]) this definition can be extended to the case in which the labels c_i are in $[k]$ for some $k \in \mathbb{N}$, in which case we obtain *multivalued* decision lists based on G. We also permit the default output to be chosen from among all elements of $[k]$ in this case (rather than having it necessarily equal 0). We define *k-$DL(G)$, the class of $[k]$-valued decision lists based on G*, to be the set of finite sequences

$$f = (f_1, c_1), (f_2, c_2), \ldots, (f_r, c_r), d$$

such that $f_i \in G$, $c_i \in [k]$ for $1 \leqslant i \leqslant r$ and d (the default value) is in $[k]$. The values of f are defined by $f(y) = c_j$ where $j = \min\{i : f_i(y) = 1\}$, or d if there are no j such that $f_j(y) = 1$. We call each f_j a *test* (or, following Krause [20], a *query*) and the pair (f_j, c_j) a *term* of the decision list.

2.2. Threshold functions and threshold decision lists

A function $t : \mathbb{R}^n \to \{0, 1\}$ is a *threshold function* if there are $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ such that

$$t(x) = \begin{cases} 1 & \text{if } \langle w, x\rangle \geqslant \theta, \\ 0 & \text{if } \langle w, x\rangle < \theta, \end{cases}$$

where $\langle w, x\rangle$ is the standard inner product of w and x. In other words, $t(x) = \mathrm{sgn}(\langle w, x\rangle - \theta)$, where $\mathrm{sgn}(z) = 1$ if $z \geqslant 0$ and $\mathrm{sgn}(z) = 0$ if $z < 0$. Given such w and θ, we say that t is represented by $[w, \theta]$ and we write $t \leftarrow [w, \theta]$. The vector w is known as the *weight-vector*, and θ is known as the *threshold*.

We now consider the class of decision lists in which the tests are threshold functions, and in which the domain is $\mathbb{R}^n$. We shall call such decision lists ($[k]$-valued) *threshold decision lists*, but they have (for the binary classification case, $k = 1$) also been called *neural* decision lists [24] and *linear* decision lists [40]. Formally, a threshold decision list

$$f = (f_1, c_1), (f_2, c_2), \ldots, (f_r, c_r), d$$

has each $f_i : \mathbb{R}^n \to \{0, 1\}$ of the form $f_i(x) = \mathrm{sgn}(\langle w_i, x\rangle - \theta_i)$ for some $w_i \in \mathbb{R}^n$ and $\theta_i \in \mathbb{R}$. The value of f on $y \in \mathbb{R}^n$ is $f(y) = c_j$ if $j = \min\{i : f_i(y) = 1\}$ exists, or it is the default value d otherwise (that is, if there are no j such that $f_j(y) = 1$).

A geometrical motivation for the use of threshold decision lists can be provided. Suppose we are given some data points in $\mathbb{R}^n$, each one of which is labeled with some member of $[k]$. We can use a hyperplane to separate off a set of points all having the same classification label. These points can then be removed from consideration and the procedure iterated until no points remain. This procedure is similar in nature to one of Jeroslow [18] (for the binary case $k = 1$), but at each stage in his procedure, only examples with label 1 may be 'chopped off' (and one cannot choose instead to chop off a set of points with label 0).

We may regard the chopping procedure as a means of constructing a threshold decision list extension of the data set. If, at stage i of the procedure, the hyperplane with equation $\sum_{i=1}^n \alpha_i y_i = \theta$ chops off points all having label j, and these lie on the side of the hyperplane with equation $\sum_{i=1}^n \alpha_i y_i > \theta$, then we take as the ith term of the threshold decision list the pair (f_i, j), where $f_i \leftarrow [\alpha, \theta]$; otherwise take the ith term to be (g_i, j), where $g_i \leftarrow [-\alpha, -\theta]$. (We may assume that no point lies on any of the defining hyperplanes.) Therefore, given any data set D (without contradictory labels), there will always be some threshold decision list extension of D. It may be that one needs to use long threshold decision lists to create an extension, in which case it might be worth considering shorter threshold decision lists which are partial extensions (thereby using a tradeoff between decision list length and goodness-of-fit to the dataset).

In the binary classification case ($k = 1$), if this construction is applied to the sequence of hyperplanes resulting from the Jeroslow method, a restricted form of decision list results, namely one in which all terms are of the form $(f_i, 1)$. But such a decision list is quite simply the *disjunction* $f_1 \vee f_2 \vee \cdots$, where $\vee$ means 'or'. For Boolean functions, the problem of decomposing a function into the disjunction of threshold functions has been considered by Hammer et al. [17] and Zuev and Lipkin [44]. Hammer et al.

defined the *threshold number* of a Boolean function to be the minimum s such that f is a disjunction of s threshold functions, and they showed that there is an increasing function with threshold number $\binom{n}{n/2}/n$. (A function is increasing if, when $f(x) = 1$ and $x_i = 0$, then $f(x + e_i) = 1$ too, where e_i is the unit basis vector with ith entry equal to 1 and all other entries equal to 0.) Zuev and Lipkin showed that almost all increasing functions have this order of threshold number, and that almost all Boolean functions have a threshold number that is $\Omega(2^n/2)$ and $O(2^n \ln n/n)$.

We give one example for illustration. This example is perhaps a little artificial as a model of a real dataset, but it involves one of the most commonly-considered Boolean functions, and also demonstrates fairly strikingly the advantages to be gained by the threshold decision list approach over the Jeroslow approach.

Example. Suppose the data set D consists of all points of $\{0, 1\}^n$, labeled according to their parity, so the (binary) classification is 1 precisely when the point has an odd number of ones. We first find a hyperplane such that all points on one side of the plane are either positive (labeled 1) or negative (labeled 0). It is clear that all we can do at this first stage is chop off one of the points since the nearest neighbors of any given point have the opposite classification. Let us suppose that we decide to chop off the origin. We may take as the first hyperplane the plane with equation $y_1 + y_2 + \cdots + y_n = 1/2$. We then ignore the origin and consider the remaining points. We can next chop off all neighbors of the origin, all the points which have precisely one entry equal to 1. All of these are positive points and the hyperplane $y_1 + y_2 + \cdots + y_n = 3/2$ will separate them from the other points. These points are then deleted from consideration. We may continue in this manner. The procedure iterates n times, and at stage i in the procedure we 'chop off' all data points having precisely $(i - 1)$ ones, by using the hyperplane $y_1 + y_2 + \cdots + y_n = i - 1/2$, for example. (These hyperplanes are in fact all parallel, but this is not in general possible.)

The decision lists arising from the chopping procedure are more general than disjunctions of threshold functions and may provide a more compact representation of the data. (That is, since fewer hyperplanes might be used, the decision list could be smaller.) Indeed, Jeroslow's method requires 2^{n-1} iterations in the parity-based example just given, since at each stage it can only 'chop off' one positive point, whereas, as we have seen, the threshold decision list approach needs only n iterations.

The chopping procedure described above suggests that the use of threshold decision lists is fairly natural, if one is to take an iterative approach to data classification. There are other methods which similarly make use of such an iterative approach, by classifying some points of the data set, removing these from consideration, and proceeding iteratively. Magasarian's multisurface method [23] (MSM) for binary classification also has this character. At each stage, it finds two parallel hyperplanes (as close together as possible) such that the points not enclosed between the two planes all have the same classification. It then removes these points and repeats. We can see that the MSM method may be regarded as constructing a decision list, where the base functions G are the indicator functions of the sets that are complements of the regions lying between two parallel hyperplanes.

The chopping procedure as we have described it is in some ways merely a device to help us see that threshold decision lists have a fairly natural geometric interpretation. But the practicalities for the binary classification case have been investigated by Marchand and Golea [24] and Marchand et al. [25], who derive a greedy heuristic for constructing a sequence of 'chops'. This relies on an incremental heuristic for the NP-hard problem of finding at each stage a hyperplane that chops off as many remaining points as possible (all of the same classification).

2.3. Multithreshold functions

We noted in the example given above that the hyperplanes of the resulting threshold decision list were parallel. By demanding that the hyperplanes are parallel, we obtain a special subclass of threshold decision lists, known as the *multithreshold functions*. These have been considered, for the binary classification case, in a number of papers, such as [13,34,39], for instance.

We define a $[k]$-*valued s-threshold function* f to be a $[k]$-valued function that is representable by a threshold decision list of length s with the test hyperplanes parallel to each other. We say that f has s *levels*. Any such function is defined by s parallel hyperplanes, which divide $\mathbb{R}^n$ into $s + 1$ regions. The function assigns points in the same region the same classification from $[k]$. Equivalently (following an observation of Bohossion and Bruck for the binary classification case [13]), f is a $[k]$-valued s-threshold function if there is a weight-vector $w = (w_1, w_2, \ldots, w_n)$ such that $f(x) = F\left(\sum_{i=1}^n w_i x_i\right)$, where the function $F : \mathbb{R} \to [k]$ is piecewise constant with at most $s + 1$ pieces. Without any loss, we may suppose that the classifications assigned to points in neighboring regions are different (for, otherwise, at least one of the planes is redundant). A special case is that of binary classification ($k = 1$), in which we may therefore assume that the classifications alternate between 0 and 1 as we traverse the regions in the direction of the normal vector common to the hyperplanes.

This method of classification is reasonably powerful. For example, for binary classification, Bohossian and Bruck observed that any Boolean function is a 2^n-level threshold function, an appropriate weight-vector being $w = (2^{n-1}, 2^{n-2}, \ldots, 2, 1)$. (For that reason, they paid particular attention to the question of whether a function can be computed by a multithreshold function where the number of levels is polynomial.)

Later in the paper, we shall want to consider a special type of $[k]$-valued k-threshold function, in which, as we traverse the regions in one of the two directions normal to the planes, the classifications take, in order, the values $0, 1, 2, \ldots, k$. We shall refer to such functions as *monotonic* multithreshold functions (and sometimes use the abbreviation k-MMTF). One reason for being interested in monotonic multithreshold functions is that they can be regarded as discretized (or finite-precision) versions of monotonic neural network activation functions. Generally, the activity of a neuron in an artificial neural network is described by an *activation function*, σ (usually an increasing function mapping into $[0, 1]$) and if the inputs to the neuron are denoted $x_1, x_2, \ldots, x_n$, then its output is given by

$$\sigma\left(\sum_{i=1}^{n} w_i x_i + w_0\right),$$

where $w_0, w_1, \ldots, w_n$ are variable parameters or weights, indicating synaptic connection strengths. For instance, an important example of an activation function is the *standard sigmoid*, $\sigma(z) = 1/(1 + e^{-z})$. Assuming σ is increasing, to 'discretize' such activity, we might replace σ by a monotonic piecewise-constant function $f : \mathbb{R}^n \to K$, where K is a finite subset of $[0, 1]$. Explicitly, suppose $|K| = k + 1$ and that the values of z at which f jumps are $z_1, z_2, \ldots, z_k$. Then the multithreshold function with weights $w_1, \ldots, w_n$ and thresholds $z_1 - w_0, z_2 - w_0, \ldots, z_k - w_0$ coincides with f if the values in K are transformed monotonically to $\{0, 1, \ldots, k\}$. Generally, in a data classification problem where the classification labels belong to a finite set $\{0, 1, \ldots, k\}$, it could be appropriate to use monotonic multithreshold functions if it is thought that the classification ought to depend monotonically on a weighted sum of the components of the data points.

Note that f is a $[k]$-valued monotonic multithreshold function if there are *weights* $w_1, w_2, \ldots, w_n$ and *thresholds* $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k$ such that, for $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, $f(x)$ is the least $r \in [k]$ such that $\sum_{i=1}^{n} w_i x_i \geqslant \theta_r$, if such an r exists; and $f(x) = 0$ otherwise.

Monotonic multithreshold functions have also proven to be of interest in multiple-valued logic. Obradović and Parberry [33], and Ngom et al. [28–30] examined special types of multithreshold functions, where the domain of the functions was taken to be $[k]^n$, rather than $\mathbb{R}^n$ as here. That is, they considered the $(k + 1)$-ary functions [33] or $(k + 1)$-*valued logic functions* [29] corresponding to k-valued multithreshold functions.

3. Generalization from random data

Recall that an extension of a labeled data set D is a function f agreeing with the classifications of the points in D, and that a partial extension is one agreeing with some (in practice, a large proportion) of the classifications in D. If a particularly simple type of extension (or a good partial extension) to a fairly large data set can be found we might expect, given the success of this simple function in explaining the large data set, that this extension will perform well on 'most' unseen data. (This is, in some senses, an instance of the 'Occam's razor' principle: we trust a simple explanation of the data.) Issues such as these have been well-studied in 'computational learning theory' and 'statistical learning theory'. (See [5,42], for instance.) To formalize the ideas somewhat, we assume that the types of extension which can be produced all belong to a particular class, H, of functions, known as the *hypothesis space*. The choice of hypothesis space might reflect either our belief about the mechanism by which the data points are labeled (for example, by some deterministic *target concept* of a particular type) or our intention only to accept simple types of explanation of the data, even if these do not match the data exactly.

We shall apply some probabilistic techniques to analyze the performance of threshold decision list classification of random data. These methods have been used in learning theory (see [6,12,42]) and originated in the work of Vapnik and Chervonenkis [43]. Following a form of the PAC model of computational learning theory, we assume that the labeled data points (x, b) (where $x \in \mathbb{R}^n$ and $b \in [k]$) have been generated randomly (perhaps from some larger corpus of data) according to a fixed probability distribution P on $Z = \mathbb{R}^n \times [k]$. (Note that this includes as a special case the situation in which x is drawn according to a fixed distribution μ on $\mathbb{R}^n$ and the label b is then given by $b = t(x)$ where t is some fixed function from $\mathbb{R}^n$ to $[k]$.) Thus, if there are m data points in D, we may regard the data set D as a vector in Z^m, drawn randomly according to the product probability distribution P^m. (This suggests that we must attach some ordering to the points, and clearly there is some ambiguity as to how to do this, but this will not turn out to be a problem for the analysis of this paper.) Given any function $f \in H$, we measure how well f extends the data set D through its *sample error*

$$\mathrm{er}_D(f) = \frac{1}{|D|}\,|\{(x, b) \in D : f(x) \neq b\}|\,.$$

This is the proportion of points of D incorrectly classified by f. We quantify how well f performs on further examples by means of its *error*

$$\mathrm{er}(f) = P\left(\{(x,b) \in Z : f(x) \neq b\}\right),$$

the probability that a randomly drawn labeled data point would be incorrectly classified by f.

What we would wish for is some guarantee that the sample error $\mathrm{er}_D(f)$ is a good approximation to the error $\mathrm{er}(f)$ for all f, so that an f with small sample error will likely have small error and therefore be a good model of the data labels. The following result provides such a guarantee for threshold decision lists and multithreshold functions of a fixed length s.

Theorem 1. *Suppose that s, n and k are fixed positive integers and that D is a data set of m labeled points (x,b) of $Z = \mathbb{R}^n \times [k]$, each generated at random according to a fixed probability distribution P on Z. Let δ be any positive number less than one. Then the following statements each hold with probability at least $1-\delta$ (where the probability is with respect to P^m, over the random choice of D):*

(1) *If f is a $[k]$-valued threshold decision list with s terms, then the error $\mathrm{er}(f)$ of f and its sample error on D, $\mathrm{er}_D(f)$ are such that $\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \varepsilon_1$, where*

$$\varepsilon_1 = \frac{4}{m}\left((s+1)\ln(2k+2) + ns\,\ln\left(\frac{2em}{n}\right) + \ln\left(\frac{4}{\delta}\right)\right).$$

(2) *If f is a $[k]$-valued s-level threshold function, then $\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \varepsilon_2$, where*

$$\varepsilon_2 = \frac{4}{m}\left((s+1)\ln(k+1) + (n+s-1)\ln\left(\frac{2ems}{n+s-1}\right) + \ln\left(\frac{4}{\delta}\right)\right).$$

The following variation of this result, in which s is not prescribed in advance, is more useful, since one does not necessarily know a priori how many terms a suitable threshold decision list will have.

Theorem 2. *Suppose that n and k are fixed positive integers and that D is a data set of m labeled points (x,b) of $Z = \mathbb{R}^n \times [k]$, each generated at random according to a fixed probability distribution P on Z. Let δ be any positive number less than one. Then the following statements each hold with probability at least $1-\delta$:*

(1) *If f is a $[k]$-valued threshold decision list, then $\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \varepsilon_1'$, where*

$$\varepsilon_1' = \frac{4}{m}\left((s+1)\ln(2k+2) + ns\ln\left(\frac{2em}{n}\right) + \ln\left(\frac{7s^2}{\delta}\right)\right),$$

where s is the number of terms of f.

(2) *If f is a $[k]$-valued multithreshold function, then $\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \varepsilon_2'$, where*

$$\varepsilon_2' = \frac{4}{m}\left((s+1)\ln(k+1) + (n+s-1)\ln\left(\frac{2ems}{n+s-1}\right) + \ln\left(\frac{7s^2}{\delta}\right)\right),$$

where s is the number of levels of f.

4. Proofs of the generalization error bounds

4.1. Bounding the error

To use results from statistical learning theory, we first need to define the *growth function* of a set of functions H mapping from $X = \mathbb{R}^n$ to $\{0,1\}$. Let $\Pi_H : \mathbb{N} \to \mathbb{N}$ be given by

$$\Pi_H(m) = \max\{|H|_S| : S \subseteq X, |S| = m\},$$

where $H|_S$ denotes H restricted to domain S and $|H|_S|$, its cardinality, is thus the number of different ways in which the points of S can (as a group) be classified by functions in H. Note that $\Pi_H(m) \leqslant 2^m$ for all m. The function Π_H is known as the growth

function of H. To obtain a sample complexity bound for the case in which the functions map into $[k]$ for $k \geqslant 2$, we use the *graphs* of the functions [12,27]. For $h \in H$, let $\mathcal{G}h$, the *graph of h*, be the function from $\mathbb{R}^n \times [k]$ to $\{0, 1\}$ defined by

$$\mathcal{G}h(x, y) = 1 \iff h(x) = y \quad (\text{for} (x, y) \in \mathbb{R}^n \times [k])$$

and let $\mathcal{G}H = \{\mathcal{G}h : h \in H\}$, the *graph space* of H. The key probability result we employ is the following bound, which follows from a result due to Vapnik [41].

Theorem 3. *Suppose that H is a set of functions from X to $[k]$ where $k \in \mathbb{N}$ and suppose that P is a probability measure on $Z = X \times [k]$. Then, for any $\eta \in (0, 1)$,*

$$P^m\left(\left\{D \in Z^m : \forall f \in H, \ \frac{\mathrm{er}(f) - \mathrm{er}_D(f)}{\sqrt{\mathrm{er}(f)}} < \eta\right\}\right) > 1 - 4\,\Pi_{\mathcal{G}H}(2m)\,\mathrm{e}^{-m\eta^2/4}.$$

Thus, we can obtain (probabilistic) bounds on the error $\mathrm{er}(f)$ of a (partial) extension from a class H when we know something about the growth function of $\mathcal{G}H$.

One consequence of this is the following general result.

Theorem 4. *Suppose H is some set of functions from a domain X into $[k]$, where $k \in \mathbb{N}$. Suppose D is a data set of m labeled points (x, b) of $Z = X \times [k]$, each generated at random according to a fixed probability distribution P on Z. Let δ be any positive number less than one. Then the following holds with probability at least $1 - \delta$: for all $f \in H$,*

$$\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \frac{4}{m}\left(\ln(\Pi_{\mathcal{G}H}(2m)) + \ln\left(\frac{4}{\delta}\right)\right).$$

Proof. Theorem 3 shows that, with probability at least $1 - \delta$, for all $f \in H$, $\mathrm{er}(f) < \mathrm{er}_D(f) + \gamma\sqrt{\mathrm{er}(f)}$, where

$$\gamma = \sqrt{\frac{4}{m}\left(\ln(\Pi_{\mathcal{G}H}(2m)) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

(We note that if η is chosen to equal γ, then the expression on the right-hand side of the inequality of Theorem 3 equals $1 - \delta$.) Now, this means

$$\mathrm{er}(f) - \gamma\sqrt{\mathrm{er}(f)} - \mathrm{er}_D(f) < 0$$

which, regarding this as a quadratic inequality in the nonnegative quantity $\sqrt{\mathrm{er}(f)}$, implies that

$$\sqrt{\mathrm{er}(f)} < \frac{\gamma}{2} + \frac{\sqrt{\gamma^2 + 4\,\mathrm{er}_D(f)}}{2}$$

and so

$$\begin{aligned}
\mathrm{er}(f) &< \left(\frac{\gamma}{2} + \frac{\sqrt{\gamma^2 + 4\,\mathrm{er}_D(f)}}{2}\right)^2 \\
&= \frac{\gamma^2}{4} + \frac{1}{4}(\gamma^2 + 4\,\mathrm{er}_D(f)) + \frac{\gamma}{2}\sqrt{\gamma^2 + 4\,\mathrm{er}_D(f)} \\
&\leqslant \frac{\gamma^2}{2} + \mathrm{er}_D(f) + \frac{1}{2}(\gamma^2 + 4\,\mathrm{er}_D(f)) \\
&= \gamma^2 + 3\,\mathrm{er}_D(f),
\end{aligned}$$

as required. □

Thus, if the growth function is suitably small, the error $\mathrm{er}(f)$ will (probably) be close to 3 times the sample error. It is possible to replace 3 by any number greater than 1, as the following result (following [5]) shows.

Theorem 5. *Suppose $\alpha > 0$. Then, with the same notations as in Theorem 4, the following holds with probability at least $1 - \delta$: for all $f \in H$,*

$$\mathrm{er}(f) < (1 + \alpha)\,\mathrm{er}_D(f) + \left(1 + \frac{1}{\alpha}\right)\frac{4}{m}\left(\ln(\Pi_{\mathcal{G}H}(2m)) + \ln\left(\frac{4}{\delta}\right)\right).$$

Proof. As noted in the proof of Theorem 4, we have, by Theorem 3, that if

$$\gamma = \sqrt{\frac{4}{m}\left(\ln(\Pi_{\mathscr{G}_H}(2m)) + \ln\left(\frac{4}{\delta}\right)\right)}$$

then, with probability at least $1-\delta$, for all $f \in H$, $\mathrm{er}(f) < \mathrm{er}_D(f) + \gamma\sqrt{\mathrm{er}(f)}$. Suppose, then, that this holds. If $\mathrm{er}(f) < (1+1/\alpha)^2\gamma^2$, then

$$\mathrm{er}(f) < \mathrm{er}_D(f) + \gamma\sqrt{\mathrm{er}(f)} < \mathrm{er}_D(f) + \gamma\sqrt{(1+1/\alpha)^2\gamma^2} < \mathrm{er}_D(f) + (1+1/\alpha)\gamma^2.$$

If, on the other hand, $\mathrm{er}(f) \geqslant (1+1/\alpha)^2\gamma^2$, then $\gamma \leqslant (\alpha/(\alpha+1))\sqrt{\mathrm{er}(f)}$ and so

$$\begin{aligned}\mathrm{er}(f) &< \mathrm{er}_D(f) + \gamma\sqrt{\mathrm{er}(f)} \\ &\leqslant \mathrm{er}_D(f) + \frac{\alpha}{\alpha+1}\sqrt{\mathrm{er}(f)}\sqrt{\mathrm{er}(f)} \\ &= \mathrm{er}_D(f) + \frac{\alpha}{\alpha+1}\,\mathrm{er}(f),\end{aligned}$$

so that $\mathrm{er}(f) < (1+\alpha)\,\mathrm{er}_D(f)$. In any case, therefore,

$$\mathrm{er}(f) < (1+\alpha)\,\mathrm{er}_D(f) + (1+1/\alpha)\gamma^2,$$

as required. □

A slightly weaker form of Theorem 4 follows on taking $\alpha = 2$. For the sake of simplicity, we have used Theorem 4 to derive the results in Section 3, but variants of those results (with choices of α other than 2) will follow on using Theorem 5.

It is possible, using a result of Vapnik and Chervonenkis [43], to obtain a result of the form: with probability at least $1-\delta$, for all $f \in H$,

$$\mathrm{er}(f) < \mathrm{er}_D(f) + \sqrt{\frac{8}{m}\left(\ln(\Pi_{\mathscr{G}_H}(2m)) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

(Such results, for binary classification by threshold decision lists, can be found in [3].) However, although this has no constant greater than 1 multiplying the sample error, the additional term is larger than in the bounds given above, since it is the square root of a number which (for large m) is less than 1. The bounds of the type presented here are therefore more useful when the sample error is small.

4.2. Growth function bounds

We now bound the growth functions of the graph spaces corresponding to the classes of interest. These results improve and extend results presented in [3] for the binary classification case. We start with general threshold decision lists. We consider the set of $[k]$-valued threshold decision lists on $\mathbb{R}^n$ with some number s of terms. (So, the length of the list is s.) We have the following bound.

Theorem 6. *Let H be the set of $[k]$-valued threshold decision lists on $\mathbb{R}^n$ with s terms, where $n, s \in \mathbb{N}$. Then*

$$\Pi_{\mathscr{G}_H}(m) \leqslant (k+1)^{s+1}s!\binom{2N}{s},$$

where $N = \sum_{i=0}^{n}\binom{m-1}{i}$.

Proof. Let S be any given set of m points in $\mathbb{R}^n$. Suppose we have two decision lists

$$f = (f_1, c_1), \ldots, (f_s, c_s), d_f, \quad g = (g_1, d_1), \ldots, (g_s, d_s), d_g$$

in H, where each f_i and g_j belong to G, the set of threshold functions on $\mathbb{R}^n$, and the labels c_i, d_j and default values d_f, d_g belong to $[k]$. Certainly, if (i) $d_f = d_g$ and $c_j = d_j$ for each j, and (ii) $f_j(x) = g_j(x)$ for all $x \in S$, then f and g are equal on S. (There are, of course, other ways in which f and g could agree on their classifications of the points of S, but for the purposes of upper-bounding the growth function, it suffices to account for only some of the possible ways in which such agreement might

occur. It is possible that better upper bounds could be obtained by a finer analysis that would take into account other ways in which two functions on H would classify S in the same way.) For fixed j, the condition in (ii) is an equivalence relation among functions in K, and the number of equivalence classes is $|G|_S|$ where G is the set of threshold functions. This is bounded by $\Pi_G(m)$, which, it is well-known [5,12,14], is bounded above as follows:

$$\Pi_G(m) \leqslant 2\sum_{i=0}^{n}\binom{m-1}{i} = 2N,$$

where N is as in the statement of the theorem. We can assume, without loss of generality, that no test is repeated in a threshold decision list (since the second occurrence of the test would be redundant, the examples that 'pass' it already having been classified). Thus, $|\mathcal{G}H|_S|$, which is no more than $|H|_S|$, can be upper bounded by the number of threshold decision lists formed from some choice of s distinct functions chosen from a set of $2N$ distinct representatives of the equivalence classes. There are $\binom{2N}{s}$ ways of choosing the tests for the list, $(k+1)^s$ ways of choosing corresponding labels, $(k+1)$ choices for the default value, and $s!$ ways of ordering the resulting tests into a decision list. Hence

$$|\mathcal{G}H|_S| \leqslant (k+1)^s(k+1)s!\binom{2N}{s},$$

as stated. □

For the binary classification case, there is a useful connection between certain types of decision list and threshold functions. A decision list defined on $\{0,1\}^n$ is a 1-*decision list* if the Boolean function in each test is given by a single literal. (So, for each i, there is some l_i such that *either* $f_i(y)=1$ if and only if $y_{l_i}=1$, *or* $f_i(y)=1$ if and only if $y_{l_i}=0$.) Then, it is known [16] (see also [2,7]) that any 1-decision list is a threshold function. Analogously, any (binary) threshold decision list is a threshold function of threshold functions [1]. But such a function is computable by a two-layer threshold network, a simple and well-studied type of artificial neural network. (A similar observation was made by Marchand et al. [24,25].) So another way of bounding the growth function of threshold decision lists in the binary case is to use some known bounds [5,9] for the growth functions of linear threshold networks. This gives a similar, though slightly looser, upper bound.

To bound the growth function of the subclass consisting of $[k]$-valued s-threshold functions, we use a result from [4], which shows that the number of ways in which a set S of m points can be partitioned by s parallel hyperplanes is at most $\sum_{i=0}^{n+s-1}\binom{sm}{i}$. (For fixed n and s, this bound is tight to within a constant, as a function of m.) Noting that we assume adjacent regions to have different classifications, to each such partition there exist at most $(k+1)k^s$ s-level threshold functions (defined on the domain restricted to S) and we therefore have the following bound.

Theorem 7. *Let H be the set of $[k]$-valued s-threshold functions on $\mathbb{R}^n$. Then*

$$\Pi_H(m) \leqslant (k+1)k^s\sum_{i=0}^{n+s-1}\binom{sm}{i}.$$

4.3. Proofs of the generalization bounds

Theorems 1 and 2 follow from Theorem 4 together with the growth function bounds just obtained and a useful approximation, which is as follows: for positive integers u, v, if $u \geqslant v$ then $\sum_{i=0}^{v}\binom{u}{i} < (eu/v)^v$. (See [5,12], for example, for a proof of this.) This inequality then implies that if H is the set of $[k]$-valued threshold decision lists of length s on $\mathbb{R}^n$, if $m > n$, and if N denotes $\sum_{i=0}^{n}\binom{m-1}{i}$, then

$$\begin{aligned}\Pi_{\mathcal{G}H}(m) &\leqslant (k+1)^{s+1}s!\binom{2N}{s} < (k+1)^{s+1}(2N)^s \\ &< (2k+2)^{s+1}\left(\left(\frac{e(m-1)}{n}\right)^n\right)^s < (2k+2)^{s+1}\left(\frac{em}{n}\right)^{ns}.\end{aligned}$$

Similarly, if H is the set of $[k]$-values s-threshold functions on $\mathbb{R}^n$, then for $m \geqslant n+s$,

$$\Pi_{\mathcal{G}H}(m) \leqslant (k+1)k^s\sum_{i=0}^{n+s-1}\binom{sm}{i} < (k+1)^{s+1}\left(\frac{ems}{n+s-1}\right)^{n+s-1}.$$

Theorem 1follows immediately from these approximations and Theorem 4, noting that the two bounds given in Theorem 1 are trivially true if $m \leqslant n$ and $m < n + s$, respectively (for in these cases, the probability bounds are larger than 1).

To obtain Theorem 2 we use a well-known technique often found in discussions of 'structural risk minimization' and model selection. (See [5,15,21,38,41], for instance.) We indicate how to obtain the first part of Theorem 2 (the proof of the second part being very similar).

For $s \in \mathbb{N}$, let H_s denote the set of $[k]$-valued threshold decision lists of length at most s. We know, by Theorem 1 that, with probability at least $1 - \delta$, $\mathrm{er}(f) < 3\,\mathrm{er}_D(f) + \varepsilon_1$ for all $f \in H$, where

$$\varepsilon_1 = \varepsilon_1(m, s, \delta) = \frac{4}{m}\left((s+1)\ln(2k+2) + ns\,\ln\left(\frac{2em}{n}\right) + \ln\left(\frac{4}{\delta}\right)\right).$$

Now let $(p_s)_{s=1}^{\infty}$ be any sequence of positive numbers such that $\sum_{s=1}^{\infty} p_s = 1$. Then, the probability that there is some $f \in H_s$ with $\mathrm{er}(f) \geqslant 3\,\mathrm{er}_D(f) + \varepsilon_1(m, s, p_s\delta)$ is less than $p_s\delta$. Therefore,

$$P^m(\{D \in Z^m : \exists\, s \in \mathbb{N}\ \exists\, f \in H_s \text{ such that } \mathrm{er}(f) \geqslant 3\,\mathrm{er}_D(f) + \varepsilon_1(m, s, p_s\delta)\}) \leqslant \sum_{s=1}^{\infty} p_s\delta = \delta. \tag{1}$$

The first part of Theorem 2 follows on taking $p_s = 6/(\pi^2 s^2)$. (It should be clear that other choices can be made, such as $p_s = 1/2^{s+1}$, for example. The type of sequence chosen can reflect a prior belief about the likelihood of there being a 'small' partial extension with low error, or can be thought of as a choice of penalty for having chosen a classifier involving a large number of hyperplanes.)

5. Consistent hypothesis finders for multithreshold functions

The generalization results presented above provide guarantees on the generalization performance of multithreshold functions that extend, or fit, the dataset well. In this section we consider how one might obtain a mulithreshold function fitting the data, given that one exists.

5.1. Finding consistent monotonic multithreshold functions

A *consistent hypothesis finder* for a set of functions is an algorithm that, on being presented with a set of points, each labeled with the value of some (unknown) function from the class, will produce some function in the class that is an extension of the dataset; that is, a function that achieves the same classifications on the points. In this section, for the class of monotonic multithreshold functions (for any fixed k), we present two consistent hypothesis finders: one is based on linear programming, and the other is an 'incremental' method arising from a procedure suggested by Obradović and Parberry [33].

Suppose we are given a sequence $(x_1, x_2, \ldots, x_m)$ of m points of $\mathbb{R}^n$, each of which has been labeled with the corresponding values $t(x_i)$ of some $[k]$-valued monotonic multithreshold function (k-MMTF) t, giving us a *sample*

$$D = ((x_1, t(x_1)), \ldots, (x_m, t(x_m)))$$

of t. Without knowing precisely the *target function* t, we might want to construct a k-MMTF *consistent* with t on the sample; that is, to produce a k-MMTF h such that $h(x_i) = t(x_i)$ for $i = 1, 2, \ldots, m$.

5.2. Using linear programming

Recall that f is a $[k]$-valued monotonic multithreshold function if there are *weights* $w_1, w_2, \ldots, w_n$ and *thresholds* $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k$ such that, for $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, if we define $S = \{0\} \cup \{r : \sum_{i=1}^{n} w_i x_i \geqslant \theta_r\}$, then $f(x) = \min S$, the least element of S. We refer to $w = (w_1, w_2, \ldots, w_n)$ as a weight vector and $\theta = (\theta_1, \ldots, \theta_k)$ as a *realizable* threshold vector (where the word realizable indicates that $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k$).

One approach to finding a k-MMTF consistent with a sample D of points labeled by a k-MMTF is to use linear programming. Suppose that, in the sample, m_i points have classification i (for $0 \leqslant i \leqslant k$) and denote these points by $x_1^{(i)}, x_2^{(i)}, \ldots, x_{m_i}^{(i)}$. Consider the following linear program, in which there are $n + k + 1$ real variables: w_i for $1 \leqslant i \leqslant n$, θ_j for $1 \leqslant j \leqslant k$, and y. Here, $w = (w_1, w_2, \ldots, w_n)$ and, for $a, b \in \mathbb{R}^n$, $\langle a, b\rangle$ denotes the inner product $a^T b = \sum_{i=1}^{n} a_i b_i$.

Maximize y subject to:

$$\langle w, x_j^{(r)} \rangle - \theta_r - y \geqslant 0 \quad (r = 1, 2, \ldots, k, \;\; j = 1, 2, \ldots, m_r),$$

$$\theta_{s+1} - \langle w, x_j^{(s)} \rangle - y \geqslant 0 \quad (s = 0, 1, \ldots, k-1, \;\; j = 1, 2, \ldots, m_s),$$

$$\begin{aligned} w_t &\geqslant -1 \quad (t = 1, 2, \ldots, n), \\ -w_t &\geqslant -1 \quad (t = 1, 2, \ldots, n), \\ \theta_u &\geqslant -1 \quad (u = 1, 2, \ldots, k), \\ -\theta_u &\geqslant -1 \quad (u = 1, 2, \ldots, k), \\ y &\geqslant 0. \end{aligned}$$

There are at most $2m + 2n + 2k$ constraints in total. Given that the sample is labeled according to the values of some k-MMTF t, the program is feasible and has a positive solution. Note that the first two sets of inequalities require a weight vector w and threshold vector θ such that the resulting k-MMTF correctly classifies the x_i and such that, additionally, the inner products 'clear' the required threshold by at least the amount y. Now, all of this is possible for some positive y, given the existence of t and the finiteness of D. Furthermore, the remaining constraints make the feasible region bounded, and, by scaling weight and threshold vectors, if necessary, it can be seen that t has a realizable weight vector and threshold vector satisfying these bounds. By solving this linear program, a consistent k-MMTF can therefore be obtained, and so we have a consistent hypothesis finder. This method can be made to run in polynomial time in the logarithmic cost model by using, for instance, Karmarkar's algorithm [19]. In particular, if (as in [29,33]) the sample points x_i are restricted to domain $[k]^n$, then the running time of the algorithm is bounded by a polynomial in $m(n+1)$, the size of the sample.

5.3. An incremental procedure

Obradović and Parberry [33] proposed an incremental algorithm for 'learning' k-MMTFs on the basis of a given sample of such a function. A slightly modified version of this algorithm was presented in [29]. These algorithms are generalizations of the well-known and well-studied perceptron learning algorithm (details of which may be found in [5,37]), which corresponds to the special case in which $k = 1$.

The algorithm in [33] maintains a *current weight vector* $w = (w_1, w_2, \ldots, w_n)$ and *threshold vector* $\theta = (\theta_1, \theta_2, \ldots, \theta_k)$, where $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k$. Together, these represent a *current hypothesis* MMTF h. On presentation of an example $x \in [k]^n$ together with its classification $t(x)$, if $h(x) = t(x)$ the algorithm does nothing, whereas if $h(x) \neq t(x)$ then the algorithm slightly alters w and one of the θ_i. The algorithm is, in this sense, *incremental*. Obradović and Parberry established a result along the lines of the classical 'perceptron convergence theorem', by proving that on any (possibly infinite) sequence of examples from $[k]^n$, each classified by some k-MMTF, t, there is an absolute bound on the number of mistakes (and hence updates) the algorithm can make (this bound depending on t). To prove this, they invoked the classical result for the perceptron.

As a consequence of the finiteness result of Obradović and Parberry, the incremental procedure can be used to construct a consistent hypothesis finder in the case where all the examples belong to $[k]^n$. For, given a finite sample $D = ((x_1, t(x_1)), \ldots, (x_m, t(x_m))$, we can cycle through these labeled examples repeatedly until no further updates will occur, at which point the current hypothesis must be consistent with the sample. We will give a direct proof that, more generally, this procedure for finding a consistent k-MMTF works when the examples can be in $\mathbb{R}^n$ and are not restricted to be in $[k]^n$ (which, as already mentioned, was the focus in [29,33]). First, we describe the consistent hypothesis finder in pseudo-code.

Algorithm L: **Incremental** k**-MMTF consistent hypothesis finder**
Input: `A sample` $D = ((x_1, t(x_1)), \ldots, (x_m, t(x_m))$ `of some` k-`MMTF` t.
Output: `Weights` $w_1, \ldots w_n$ `and thresholds` $\theta_1, \ldots, \theta_k$.
for all i, `set` $w_i := 0$
for all l, `set` $\theta_l := 0$
repeat `until no updates needed in a complete cycle through` D
for $i := 1$ `to` m **do**
  let h `be the current hypothesis, represented by` w `and` θ
  if $v = h(x_i) \neq t(x_i)$ **then**
    let $\Delta = t(x_i) - h(x_i) = t(x_i) - v$
    if $\Delta < 0$ `then`
      update weights and thresholds as follows

$\theta_v \leftarrow \theta_v + 1$
$\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
$w \leftarrow w - x_i$
if $\Delta > 0$ then
update weights and thresholds as follows
$\theta_{v+1} \leftarrow \theta_{v+1} - 1$
$\theta_l \leftarrow \theta_l$ for $l \neq v+1$ (i.e., no change)
$w \leftarrow w + x_i$
return $w_1, w_2, \ldots, w_n$ and $\theta_1, \ldots, \theta_k$.
end

The k-MMTF corresponding to the weights and thresholds output by the algorithm is called the *output hypothesis* of the algorithm, and is denoted $L(D)$.

Ngom et al. [29] considered a slight variant of the procedure suggested by Obradović and Parberry. (The problem they considered was slightly more general too: they were interested in incrementally learning 'permutably homogeneous perceptrons', of which k-MMTFs are a special type.) Following their variation of [33], an alternative consistent hypothesis finder can be devised that has the following update rule:

if $v = h(x_i) \neq t(x_i)$ **then**
let $\Delta = t(x_i) - h(x_i) = t(x_i) - v$
if $\Delta < 0$ then
update the weights and thresholds as follows
$\theta_v \leftarrow \theta_v + \Delta = \theta_v - |\Delta|$
$\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
$w \leftarrow w - \Delta x_i = w + |\Delta| x_i$
if $\Delta > 0$ then
update the weights and thresholds as follows
$\theta_{v+1} \leftarrow \theta_{v+1} - \Delta$
$\theta_l \leftarrow \theta_l$ for $l \neq v+1$ (i.e., no change)
$w \leftarrow w + \Delta x_i$

Thus, in this case, the extent by which the weights and thresholds are changed depends on how far $h(x_i)$ is from $t(x_i)$ and not merely on the 'sign' of the difference. A further possible modification is to have a 'learning rate' (possibly changing in time) multiplying the additive changes.

Obradović and Parberry [33] also investigated the performance of an alternative procedure in which weights and thresholds are updated *multiplicatively* rather than *additively*, following Littlestone's 'Winnow' generalization of the standard perceptron learning algorithm [22]. (Indeed, this is the primary focus of their paper.) They show that this multiplicative algorithm is in many cases better than the additive one, in that the bound on the number of updates required can be significantly smaller. This multiplicative algorithm can also, in an analogous way, be turned into a consistent hypothesis finder.

We now give a direct proof that the incremental consistent hypothesis finder works.

Theorem 8. *Given any sample D of a k-MMTF, the incremental consistent hypothesis finder for k-MLTs will terminate to produce an output hypothesis $L(D)$ consistent with D. Furthermore, if the examples x_i in the sample satisfy $\| x_i \| \leqslant R$ and if the k-MMTF t by which the sample points are labeled is represented by weight vector W and threshold vector Θ with the property that $\| W\|^2 + \| \Theta\|^2 = 1$ and no x_i lies on any of the k hyperplanes defined by W and Θ, then the total number of updates (and hence cycles) required by L is at most $(R^2+1)/\gamma^2$ where*

$$\gamma = \min\{|\langle W, x_i\rangle - \Theta_l| : 1 \leqslant i \leqslant m,\ 1 \leqslant l \leqslant k\} > 0.$$

Proof. The proof is a variant of the proof of the perceptron convergence theorem [5,31,32,37]. Clearly, since the sample is finite, there will be a weight vector W and a threshold vector Θ representing t such that no point of the sample lies on any of the k hyperplanes (because there is flexibility to perturb the thresholds). By scaling the weights and thresholds if necessary, we can further assume that $\| W\|^2 + \| \Theta\|^2 = 1$. Let W and Θ be a fixed choice of such vectors. Denote by $w(u)$ and $\theta(u)$ the weight and threshold vectors after u updates have been made, and let the components of these be $w_i(u)$ and $\theta_l(u)$. (Note that $w(0)$ and $\theta(0)$ are the all-zero vectors.) For some x in the sample, the uth update rule takes the form

$$w(u) = w(u-1) + \delta x,$$
$$\theta_l(u) = \theta_l(u-1) - \delta.$$

Here, δ is 1 or -1, according to whether $t(x) > h(x)$ or $t(x) < h(x)$, respectively; and l is, correspondingly, $v+1$ or v, where $v = h(x)$ and h is the current hypothesis (represented by $w(u-1)$ and $\theta(u-1)$). Let $N(u)$ be defined as

$$N(u) = \langle (W, \Theta), (w(u), \theta(u)) \rangle = \langle W, w(u) \rangle + \langle \Theta, \theta(u) \rangle.$$

Then,

$$N(u) - N(u-1) = \langle W, \delta x \rangle - \Theta_l \delta = \delta(\langle W, x \rangle - \Theta_l).$$

Now, if $\delta = 1$, then $l = v+1$ and, since $v = h(x) < t(x)$, we have $t(x) \geqslant v+1$ and so $\langle W, x \rangle \geqslant \Theta_{v+1} + \gamma$, as a consequence of which $\delta(\langle W, x \rangle - \Theta_{v+1}) \geqslant \gamma$. If, however, $\delta = -1$, then $l = v$ and $t(x) < h(x) = v$, so that $\langle W, x \rangle < \Theta_v - \gamma$ and hence $\delta(\langle W, x \rangle - \Theta_v) \geqslant \gamma$. In both cases, therefore, $N(u) - N(u-1) \geqslant \gamma$. It follows that $N(u) \geqslant N(0) + \gamma u = \gamma u$.

Now let

$$L(u) = \| (w(u), \theta(u)) \|^2 = \| w(u) \|^2 + \| \theta(u) \|^2.$$

From the fact that $N(u) = \langle W, w(u) \rangle + \langle \Theta, \theta(u) \rangle \geqslant u\gamma$, together with the Cauchy–Schwarz inequality and the fact that $\| (W, \Theta) \| = 1$, we have

$$\begin{aligned} L(u) &= \| w(u) \|^2 + \| \theta(u) \|^2 \\ &= \| (w(u), \theta(u)) \|^2 \\ &= \| (w(u), \theta(u)) \|^2 \, \| (W, \Theta) \|^2 \\ &\geqslant (\langle (w(u), \theta(u)), (W, \Theta) \rangle)^2 \\ &= (N(u))^2 \\ &\geqslant (\gamma u)^2. \end{aligned}$$

But, if e_l denotes the vector with lth entry equal to 1 and all other entries 0, then

$$\begin{aligned} L(u) &= \| w(u) \|^2 + \| \theta(u) \|^2 \\ &= \| w(u-1) + \delta x \|^2 + \| \theta(u-1) - \delta e_l \|^2 \\ &= \| w(u-1) \|^2 + \| \theta(u-1) \|^2 + \delta^2 \| x \|^2 + \delta^2 + 2\delta \langle w(u-1), x \rangle - 2\delta \langle \theta(u-1), e_l \rangle \\ &\leqslant L(u-1) + (R^2 + 1) + 2\delta \left(\langle w(u-1), x \rangle - \theta_l(u-1) \right). \end{aligned}$$

Because of the update rule, either $v = h(x) > t(x)$, in which case

$$\delta < 0, \;\; l = v \quad \text{and} \quad \langle w(u-1), x \rangle \geqslant \theta_v(u-1);$$

or $v = h(x) < t(x)$, and

$$\delta > 0, \quad l = v+1 \quad \text{and} \quad \langle w(u-1), x \rangle < \theta_{v+1}(u-1).$$

So, in both cases, $\delta \left(\langle w(u-1), x \rangle - \theta_l(u-1) \right) \leqslant 0$, and hence

$$L(u) \leqslant L(u-1) + (R^2 + 1)$$

and so $L(u) \leqslant L(0) + (R^2+1)u = (R^2+1)u$. It follows that

$$(\gamma u)^2 \leqslant L(u) \leqslant (R^2+1)u$$

and so $u \leqslant (R^2+1)/\gamma^2$, completing the proof. □

Note that the upper bound given in Theorem 8 on the number of updates depends both on t and on the precise points in the sample (through the dependence on γ). For the standard perceptron (the case $k = 1$), the case in which only Boolean points (that is, points of $\{0,1\}^n$) have been considered has been of particular interest historically. A counterpart to this in the case $k > 1$ is to consider only points of $[k]^n$ (as in [29,33]). With this restriction to a finite domain, for a given t, the parameter γ can of course be bounded below independently of the sample. Thus, one can bound the number of updates (and hence cycles) independently of the sample. It is, however, well-known (in the case $k = 1$) that this bound can be exponential in n; see [8,26], for instance.

This consistent hypothesis finder has an appealing on-line, incremental character, but (unlike the method based on linear programming) it is not efficient. Even when the sample points are restricted to $\{0,1\}^n$, the time taken to produce a consistent

hypothesis will not generally be polynomial in $m(n+1)$, the size of the input. For, when $k=1$, the algorithm is equivalent to the consistent hypothesis finder based on the standard perceptron learning algorithm, and this is known not to be efficient [8]. (There is a Boolean threshold function t and a set S of $n+1$ examples with the property that the only threshold function consistent with t on S is t itself and, moreover, the ratio of the largest to the smallest weight in any weight vector representing t is exponential in n. On presentation of the sample corresponding to S and t, the algorithm will necessarily make an exponential number of updates to achieve the exponential separation between the largest and smallest weights for most choices of initial weights and thresholds.)

5.4. Some open questions in the non-monotonic case

We now consider the problem of finding consistent hypotheses for the general class of multithreshold functions, where monotonicity cannot be assumed. We focus on the binary classification case (the general $[k]$-valued case being at least as difficult in the absence of monotonicity). The fact that, in this case (as noted earlier), the classifications alternate between 0 and 1, rather than take, successively, the values 0 to k makes it difficult to adapt the linear programming and incremental approaches to finding consistent multithreshold functions. Here, we raise two open questions: first, whether a procedure based on a technique proposed by Takiyama [39] is effective; and, secondly, whether there is any *efficient* means of finding a consistent hypothesis, in the restricted problem where all examples are assumed to be binary vectors.

5.4.1. An incremental method based on a procedure of Takiyama

An incremental 'learning' algorithm was proposed by Takiyama [39]. Although he cites some experimental success with the method, no analysis of its general effectiveness was undertaken. The presentation of the method in [39] is very complex, but the procedure can be described much more simply. We describe here the modification of Takiyama's method that would be applicable to cycling through a finite sample (in the hope of creating a consistent hypothesis). Suppose that a sample D for some binary-valued multithreshold function t is given. Then the procedure can be described as follows. The numbers $a(u)$ for $u \in \mathbb{N}$ constitute some prescribed sequence of positive 'learning rates', and the variable u indexes the updates made by the algorithm.

Input: Sample $D = ((x_1, t(x_1)), \ldots, (x_m, t(x_m)))$ of some $\{0,1\}$-valued k-threshold function t.
Output: Weights $w_1, \ldots, w_n$ and thresholds $\theta_1 \leqslant \theta_2 \leqslant \cdots \leqslant \theta_k$
for all i, choose w_i randomly
for all l, choose θ_l randomly, satisfying $\theta_1 \leqslant \cdots \leqslant \theta_k$
repeat until no updates needed in a complete cycle through D
for $i := 1$ to m **do**
  let h be the current hypothesis, represented by w and θ
  if $h(x_i) \neq t(x_i)$ **then**
    if $\langle w, x\rangle \leqslant (\theta_1 + \theta_2)/2$, then **let** $v = 1$
    if $\langle w, x\rangle \geqslant (\theta_{k-1} + \theta_k)/2$, then **let** $v = k$
    otherwise **let** $v \in \{2, 3, \ldots, k-1\}$ be such that
      $(\theta_{v-1} + \theta_v)/2 < \langle w, x\rangle \leqslant (\theta_v + \theta_{v+1})/2$
    if $\langle w, x\rangle \geqslant \theta_v$ **then**
    update weights and thresholds as follows
      $\theta_v \leftarrow \theta_v + a(u)$
      $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
      $w \leftarrow w - a(u)x_i$
    if $\langle w, x\rangle < \theta_v$ **then**
      update weights and thresholds as follows
      $\theta_v \leftarrow \theta_v - a(u)$
      $\theta_l \leftarrow \theta_l$ for $l \neq v$ (i.e., no change)
      $w \leftarrow w + a(u)x_i$
return $w_1, w_2, \ldots, w_n$ and $\theta_1, \ldots, \theta_k$.
end

Thus, given an example x_i misclassified by the current hypothesis (represented by w and θ), the procedure shifts the threshold θ_v nearest in value to $\langle w, x_i\rangle$ and alters the weight vector w, so as to make the quantity $\langle w, x_i\rangle - \theta_v$ decrease or increase, as appropriate.

For $k \geqslant 2$, the proof of Theorem 8 apparently cannot be adapted to show that this procedure terminates (because the fact that $h(x) \neq t(x)$ does not imply either that $\langle w, x \rangle$ is too large or that it is too small). We therefore raise the following open question:

Question: Does this procedure terminate for some choice of sequence $(a(u))_{u=1}^{\infty}$? That is, is it a consistent hypothesis finder?

5.4.2. Computational complexity of finding a consistent hypothesis

Let us suppose that the domain is restricted to $\{0, 1\}^n$, so that we are considering the *Boolean* k-threshold functions (simply referred to as Boolean threshold functions in the case $k = 1$). Even if the incremental method described above is indeed, in this case, a consistent hypothesis finder, it is not an *efficient* one, in the sense that the running time will not generally be polynomial in $m(n+1)$, the size of the input. That this is the case follows from the observation that when $k = 1$, the procedure is equivalent to the consistent hypothesis finder based on the standard perceptron algorithm, and, as already noted, this is not a polynomial-time algorithm.

The question arises therefore whether there can be some other efficient consistent hypothesis finder for the class of Boolean k-threshold functions. In the case $k = 1$, there is. For, in this case the consistent hypothesis finder based on linear programming is a consistent hypothesis finder for Boolean threshold functions, and is known to be efficient (by using, for example, Karmarkar's algorithm [19], as noted in [12]).

More specifically, consider the case $k = 2$ (and, again, Boolean domain $\{0, 1\}^n$). Corresponding to the problem of finding a consistent hypothesis, we have the following 'consistency problem':

TWO PARALLEL PLANE SEPARABILITY
Instance: $S^+ \subseteq \{0, 1\}^n$ and $S^- \subseteq \{0, 1\}^n$.
Question: Are there two parallel hyperplanes such that all points of S^+ lie between the hyperplanes and all points of S^- do not?

Assuming P $\neq$ NP, if this is an NP-complete problem, then there can certainly be no efficient consistent hypothesis finder for 2-threshold functions. Furthermore, since a class of Boolean functions is learnable in the standard PAC model of learning if and only if the corresponding consistency problem is in RP (as shown in [35], for example), unless TWO PARALLEL PLANE SEPARABILITY is in RP, there can be no efficient PAC learning algorithm for Boolean 2-threshold functions (unless P = RP). We therefore raise the following question:

Question: Is TWO PARALLEL PLANE SEPARABILITY NP-complete?

That the answer to this question could be 'yes' might be suggested by the fact that the consistency problem for the intersection of two halfspaces is NP-complete [11] (though, here, of course, the halfspaces need not necessarily be defined by parallel hyperplanes). On the other hand, for the case of single-plane separability, as already noted, the consistency problem can be solved in polynomial time.

Acknowledgements

This work was initiated during a visit to RUTCOR and DIMACS, Rutgers University, in March 2002. I am grateful to Peter Hammer and colleagues for their hospitality and stimulating discussion.

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

References

[1] M. Anthony, Threshold functions, decision lists, and the representation of Boolean functions, Neurocolt Technical Report NC-TR-96-028, 1996.
[2] M. Anthony, Discrete Mathematics of Neural Networks: Selected Topics, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001.
[3] M. Anthony, Generalization error bounds for threshold decision lists, J. Mach. Learn. Res. 5 (2004) 189–217.
[4] M. Anthony, Partitioning points by parallel planes, RUTCOR Research Report RRR-39-2002, Rutgers Center for Operations Research (Also, CDAM research report LSE-CDAM-2002-10, Centre for Discrete and Applicable Mathematics, London School of Economics), Discrete Math., 282 (2004) 17–21.
[5] M. Anthony, P.L. Bartlett, Neural Network Learning: Theoretical Foundations, Cambridge University Press, Cambridge, 1999.
[6] M. Anthony, N.L. Biggs, Computational Learning Theory: An Introduction, Cambridge Tracts in Theoretical Computer Science, Vol. 30, Cambridge University Press, Cambridge, UK, 1992.

[7] M. Anthony, G. Brightwell, J. Shawe-Taylor, On specifying Boolean functions by labelled examples, Discrete Appl. Math. 61 (1995) 1–25.
[8] M. Anthony, J. Shawe-Taylor, Using the perceptron algorithm to find consistent hypotheses, Combinatorics Probab. Comput. 4 (2) (1993) 385–387.
[9] E. Baum, D. Haussler, What size net gives valid generalization?, Neural Comput. 1 (1) (1989) 151–160.
[10] J.C. Bioch, Dualization decision lists and identification of monotone discrete functions, Ann. Math. Artificial Intell. 24 (1998) 69–91.
[11] A. Blum, R.L. Rivest, Training a 3-node neural network is NP-complete, Neural Networks 5 (1) (1992) 117–127.
[12] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth, Learnability and the Vapnik–Chervonenkis dimension, J. ACM 36 (4) (1989) 929–965.
[13] V. Bohossian, J. Bruck, Multiple threshold neural logic, M. Jordan, M. Kearns, S. Solla(Eds.), Advances in Neural Information Processing, Vol. 10, NIPS'1997, MIT Press, Cambridge, 1998.
[14] T.M. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, IEEE Trans. Electron. Comput. 14 (1965) 326–334.
[15] L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer, Berlin, 1996.
[16] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant, A general lower bound on the number of examples needed for learning, Inform. Comput. 82 (1989) 247–261.
[17] P.L. Hammer, T. Ibaraki, U.N. Peled, Threshold numbers and threshold completions, Ann. Discrete Math. 11 (1981) 125–145.
[18] R.G. Jeroslow, On defining sets of vertices of the hypercube by linear inequalities, Discrete Math. 11 (1975) 119–124.
[19] N. Karmarkar, A new polynomial time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
[20] M. Krause, On the computational power of Boolean decision lists, in: Proceedings of the 19th Annual Symposium of Theoretical Aspects of Computer Science (STACS),2002.
[21] N. Linial, Y. Mansour, R.L. Rivest, Results on learnability and the Vapnik–Chervonenkis dimension, Inform. Comput. 90 (1) (1991) 33–49.
[22] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear threshold learning algorithm, Mach. Learn. 2 (4) (1988) 245–318.
[23] O.L. Mangasarian, Multisurface method of pattern separation, IEEE Trans. Inform. Theory IT-14 (6) (1968) 801–807.
[24] M. Marchand, M. Golea, On learning simple neural concepts: from halfspace intersections to neural decision lists, Network: Comput. Neural Systems 4 (1993) 67–85.
[25] M. Marchand, M. Golea, P. Ruján, A convergence theorem for sequential learning in two-layer perceptrons, Europhys. Lett. 11 (1990) 487.
[26] M. Minsky, S. Papert, Perceptrons, MIT Press, Cambridge, MA, 1969 (Expanded edition 1988).
[27] B.K. Natarajan, On learning sets and functions, Mach. Learn. 4 (1) (1989) 67–97.
[28] A. Ngom, A. Obradović, I. Stojmenović, Minimization of multiple-valued multiple-threshold perceptrons using genetic algorithms, in: Proceedings of the 28th IEEE International Symposium on Multiple-Valued Logic, IEEE Press, New York, 1998.
[29] A. Ngom, C. Reischer, D. Simovici, I. Stojmenović, Learning with permutably homogeneous multiple-valued multiple-threshold perceptrons, in: Proceedings of the 28th IEEE International Symposium on Multiple-Valued Logic, IEEE Press, New York, 1998.
[30] A. Ngom, I. Stojmenović, J. Žunić, On the number of multilinear partitions and the computing capacity of multiple-valued multiple-threshold perceptrons, in: Proceedings of 29th IEEE International Symposium on Multiple-Valued Logic, IEEE Press, New York, 1999.
[31] N.J. Nilsson, Learning Machines, McGraw-Hill, New York, 1965.
[32] A.B. Novikoff, On convergence proofs on perceptrons, Symposium on the Mathematical Theory of Automata, Vol. 12, Polytechnic Institute of Brooklyn 1962, pp. 615–622.
[33] Z. Obradović, I. Parberry, Learning with discrete multivalued neurons, J. Comput. System Sci. 49 (1994) 375–390.
[34] S. Olafsson, Y.S. Abu-Mostafa, The capacity of multilevel threshold functions, IEEE Trans. Pattern Anal. Mach. Intell. 10 (2) (1988) 277–281.
[35] L. Pitt, L. Valiant, Computational limitations on learning from examples, J. ACM 35 (1988) 965–984.
[36] R. Rivest, Learning decision lists, Mach. Learn. 2 (3) (1987) 229–246.
[37] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (1958) 386–407.
[38] J. Shawe-Taylor, P. Bartlett, R.C. Williamson, M. Anthony, Structural risk minimisation over data-dependent hierarchies, IEEE Trans. Inform. Theory 44 (5) (1998) 1926–1940.
[39] R. Takiyama, The separating capacity of a multi-threshold element, IEEE Trans. Pattern Anal. Mach. Intell. 7 (1985) 112–116.
[40] G. Turán, F. Vatan, Linear decision lists and partitioning algorithms for the construction of neural networks, in: Foundations of Computational Mathematics selected papers of a conference,Springer, Rio de Janeiro, 1997, pp. 414–423.
[41] V.N. Vapnik, Estimation of Dependences Based on Empirical Data, Springer, New York, 1982.
[42] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[43] V.N. Vapnik, A.Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory Probab. Appl. 16 (2) (1971) 264–280.
[44] A. Zuev, L.I. Lipkin, Estimating the efficiency of threshold representations of Boolean functions, Cybernetics 24 (1988) 713–723 (translated from Kibernetika (Kiev) 6 (1988) 29–37).